Requested Patent: GB2350213A

Title: WEB PAGE DOWNLOADING

Abstracted Patent: GB2350213

Publication Date: 2000-11-22

Inventor(s):

ROONEY NOEL (GB); HUTCHISON GORDON DOUGLAS (GB)

Applicant(s): IBM (US)

Application Number: GB19990011739 19990521

Priority Number(s): GB19990011739 19990521

IPC Classification: G06F17/30

Equivalents:

ABSTRACT:

(12) UK Patent Application (19) GB (11) 2 350 213 (13) A

(43) Date of A Publication 22.11.2000

(21) Application No 9911739.2

(22) Date of Filing 21.05.1999

(51) INT CL[7]
G06F 17/30

(52) UK CL (Edition R)
G4A AMU AUDB

(56) Documents Cited
WO 97/46955 A1
http://libsun1.jr2.ox.ac.uk/training/netscape.html
HCLU,"Netscape Navigator", 1996

(58) Field of Search
UK CL (Edition R) G4A AMU AUDB
INT CL[7] G06F 17/30
ONLINE: COMPUTER EPODOC JAPIO WPI INTERNET

(71) Applicant(s)
International Business Machines Corporation
(Incorporated in USA - New York)
Armonk, New York 10504, United States of America

(72) Inventor(s)
Gordon Douglas Hutchison
Noel Rooney

(74) Agent and/or Address for Service
P Waldner
IBM United Kingdom Limited, Intellectual Property Department, Hursley Park, WINCHESTER, Hampshire, SO21 2JN, United Kingdom

(54) Abstract Title
Web page downloading

(57) A method of using a "markup language", such as HTML, browser, in which a page is loaded into display memory, the user selects multiple URLs from those displayed on the page, and the browser preloads pages corresponding to the URLs into a cache without necessarily displaying them, so the user may continue to view the original page. Before preloading a given page, it may update a navigation list with the corresponding URL, and a user may select an entry in this list, directly or via "forward" and "back" GUI buttons, to display immediately.

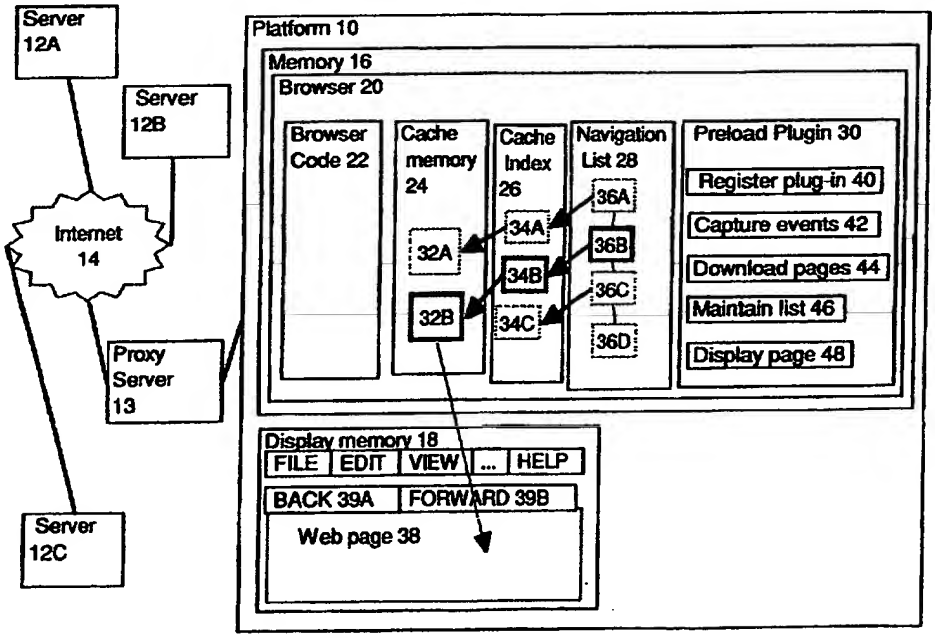

Figure 1

GB 2 350 213 A

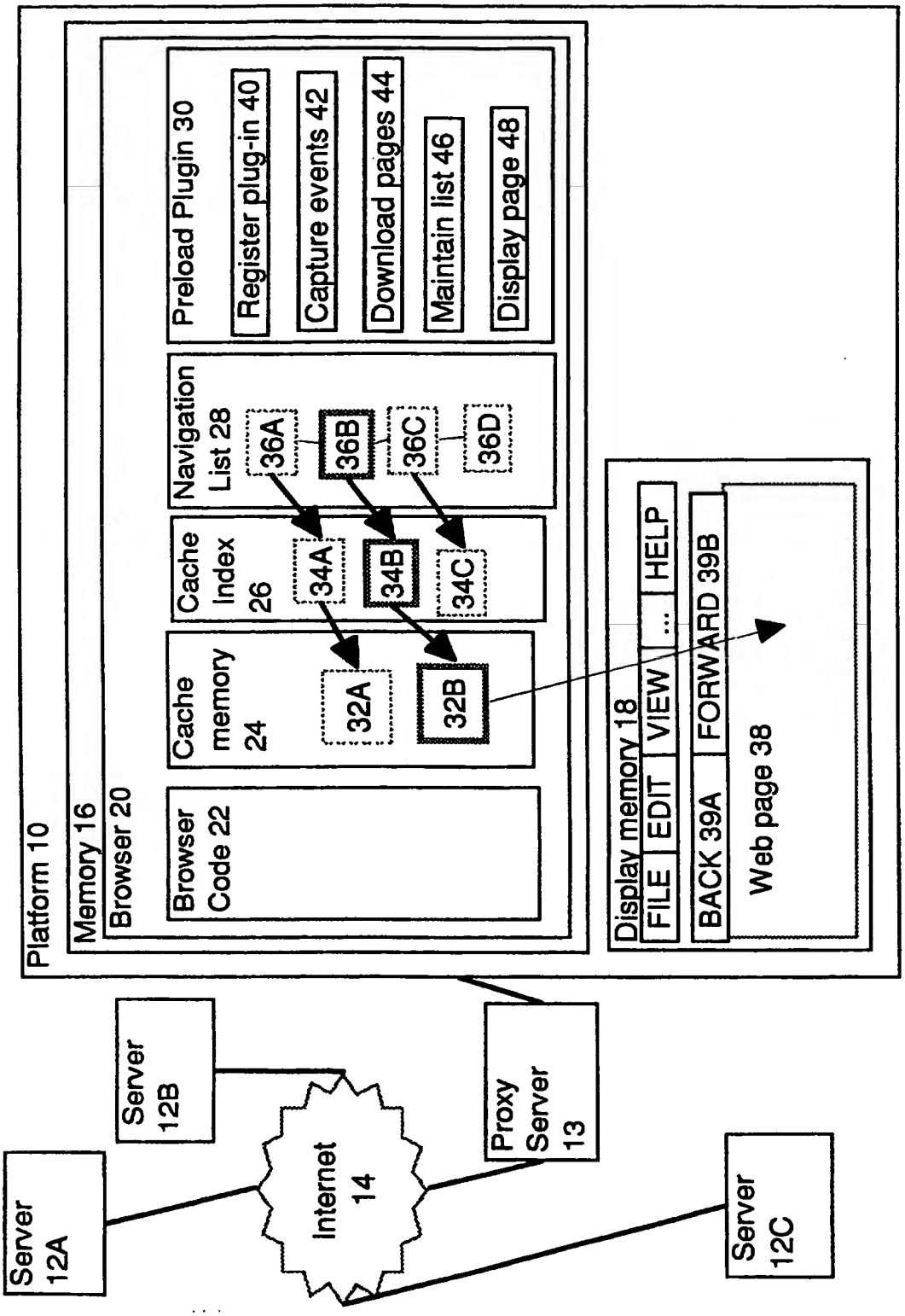
Platform 10
Memory 16
Browser 20
Browser Code 22
Cache memory 24
32A
32B
Cache Index 26
34A
34B
34C
Navigation List 28
36A
36B
36C
36D
Preload Plugin 30
Register plug-in 40
Capture events 42
Download pages 44
Maintain list 46
Display page 48
Display memory 18
FILE
EDIT
VIEW
...
HELP
BACK 39A
FORWARD 39B
Web page 38
Server 12A
Server 12B
Internet 14
Proxy Server 13
Server 12C

Figure 1

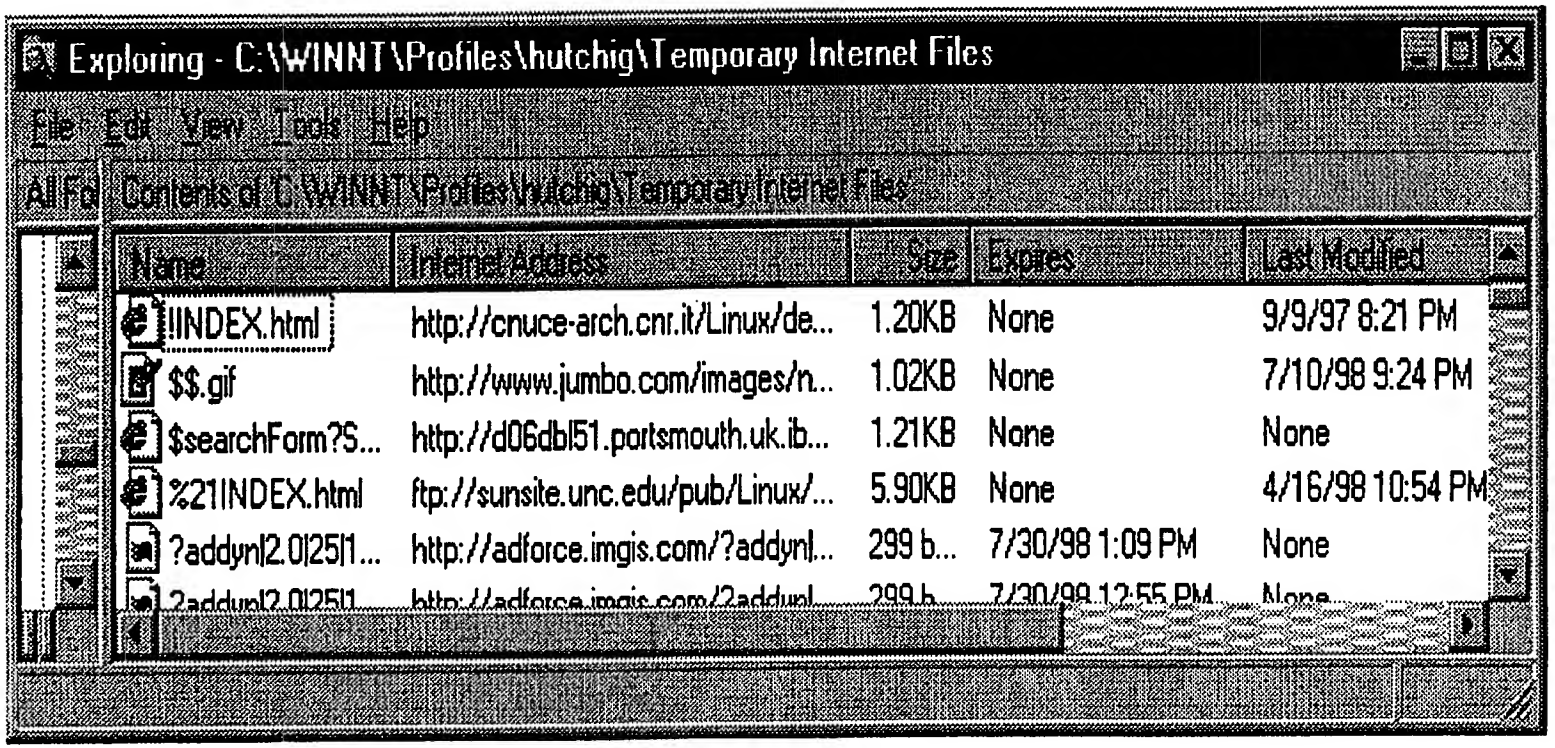
Exploring - C:\WINNT\Profiles\hutchig\Temporary Internet Files
File Edit View Tools Help
All Fo
Contents of 'C:\WINNT\Profiles\hutchig\Temporary Internet Files'
Name
Internet Address
Size
Expires
Last Modified
!INDEX.html
http://cnuce-arch.cnr.it/Linux/de...
1.20KB
None
9/9/97 8:21 PM
$$.gif
http://www.jumbo.com/images/n...
1.02KB
None
7/10/98 9:24 PM
$searchForm?S...
http://d06dbl51.portsmouth.uk.ib...
1.21KB
None
None
%21INDEX.html
ftp://sunsite.unc.edu/pub/Linux/...
5.90KB
None
4/16/98 10:54 PM
?addynl2.0|25|1...
http://adforce.imgis.com/?addynl...
299 b...
7/30/98 1:09 PM
None

Figure 2

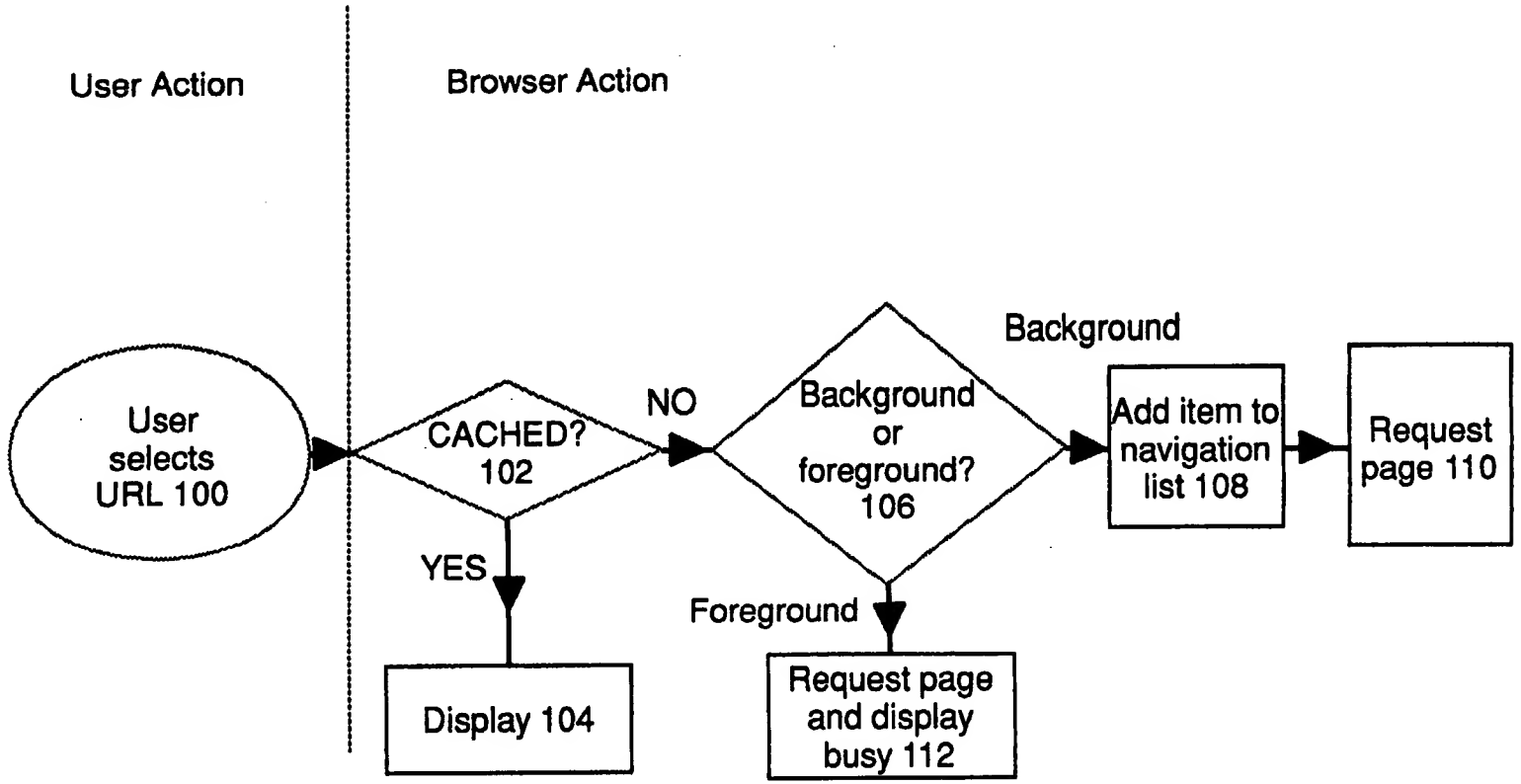
User Action
Browser Action
User selects URL 100
CACHED? 102
NO
YES
Display 104
Background or foreground? 106
Background
Foreground
Add item to navigation list 108
Request page 110
Request page and display busy 112

Figure 3A

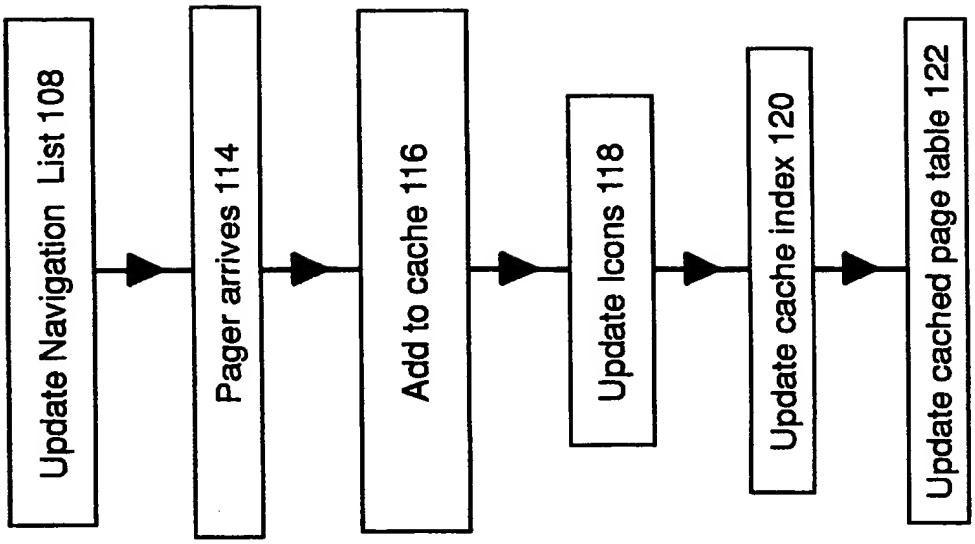
Update Navigation List 108
Pager arrives 114
Add to cache 116
Update Icons 118
Update cache index 120
Update cached page table 122

Figure 3B

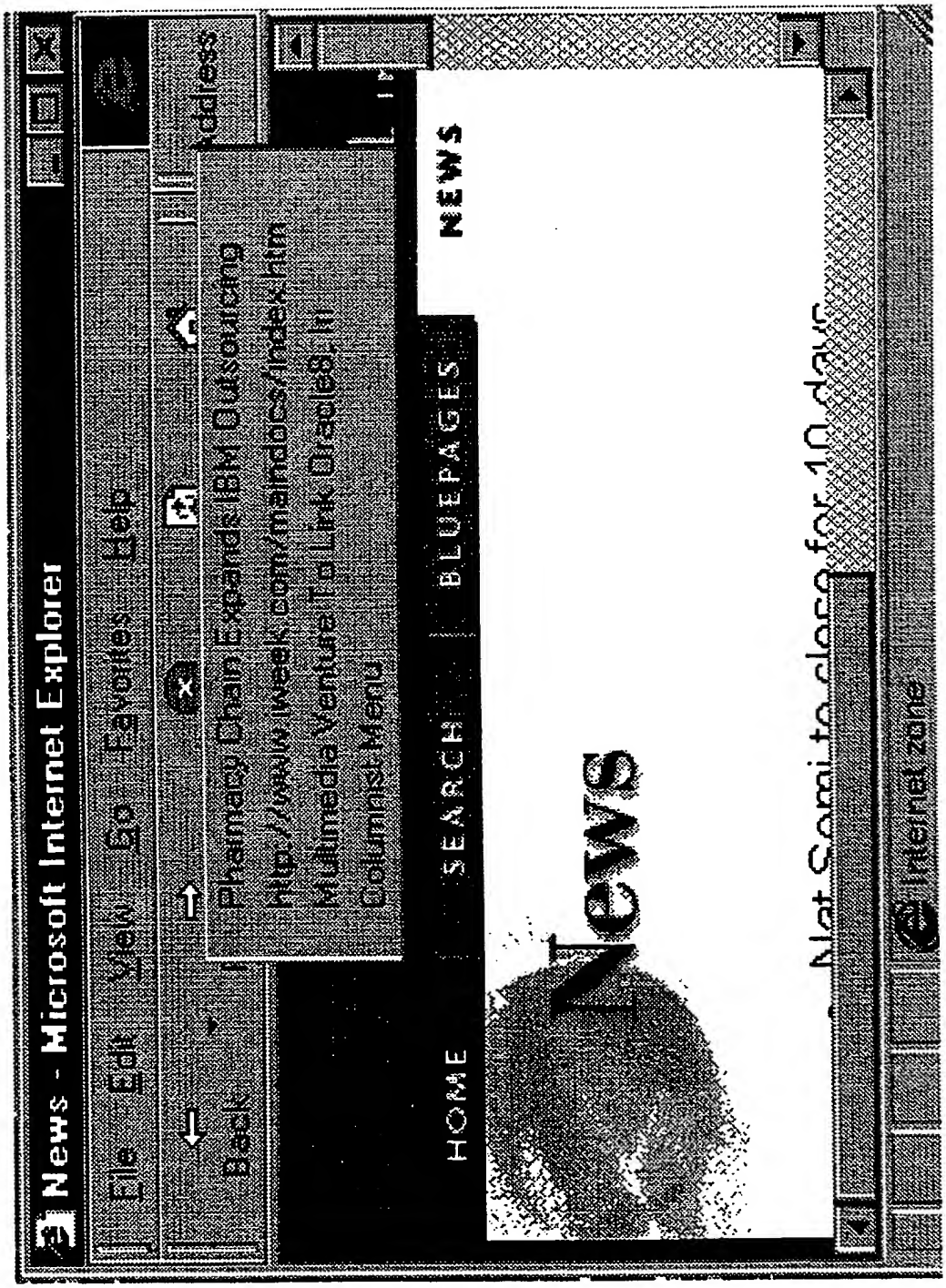
News - Microsoft Internet Explorer
File Edit View Go Favorites Help
Back
Address
Pharmacy Chain Expands IBM Outsourcing
http://www.week.com/mindocs/index.htm
Multimedia Ventura To Link Oracle8 i
Columnist Meou
HOME
SEARCH
BLUEPAGES
NEWS
News
Internet zone

Figure 4

WEB PAGE DOWNLOADING

FIELD OF INVENTION

This invention relates to the Internet world wide web page downloading and in particular relates to browser prefetching of web pages.

BACKGROUND OF INVENTION

The Internet is a large number of computers connected in an ad hoc fashion forming a world wide network. It was originally conceived by the US government to allow a scaleable network of computers to talk to each other and share information and has the added effect of being resilient to malfunction of single computers in the network. This means a lower risk of loosing total communication between computers in the event of an intermediate computer failing because the network may be connected in other ways. The communication language of the Internet is the double layer Transmission Control Protocol/Internet Protocol (TCP/IP). TCP controls the way that data is broken down into small packets for transmission over the Internet and controls the way the packets are subsequently built up in the correct order. The Internet protocol controls the addressing of the packets so that they are transmitted to and received by the correct computer.

The Internet's best known medium is the world wide web (WWW) which is very broadly a set of protocols allowing transfer of multimedia data between computers. One of these protocols is Hypertext Transfer Protocol (HTTP) which allows pages of data in hypertext mark up language (HTML) to be transferred. The application which sits on a computer to make this transfer possible is an Internet browser such as Microsoft Internet Explorer or Netscape Navigator.

A very common mode of operation in web browsers is the following. A user enters criteria into a search engine prompt or goes to an index/contents page of link. The user is then presented with list of available web Uniform Resource Locators (URLs) a subset of which he/she wishes to browse and selects one location for download. The user then waits for the page to load from the Internet server to the proxy server and to his machine before viewing the page. After finding something of interest or not the user moves back to the index/list and makes a next selection, waits for download etc. This process repeats until the user has read or collected the relevant material. In this mode of operation it is necessary to repeatedly wait, sometimes for excessive periods, for the pages to download. One solution is to launch a new browser to request a download from the selection and return to the original browser to

launch further browsers for each desired URL. This is a messy and irritating way to operate as one must keep track of where one is in the desired list and has no visual clue as to when the pages have been downloaded to the multiple windows and which to which window on the screen a particular page will appear at.

GotIt from Ahead Software Inc., Redmond, WA (www.goahead.com) preloads web pages into proxy cache. GotIt is a www proxy gateway to the Internet for providing a portal and location for fire walls, security policy and web page caching as standard. For example most www proxies are likely to fetch www.yahoo.com many times each hour and the advantage of storing it and refreshing it say only every 10 minutes is clear. GotIt allows preloads of pages linked to the current page. The proxy server will download from the servers in the background all the pages that are linked to from a page currently being viewed it then stores these on the disk of the machine running the proxy.

However in the GotIt solution the proxy has no interaction with the browser interface and visa versa during the background downloading process. This has two disadvantages: 1) The user's model is not improved since the process of moving to proxy pages remains inflexible. 2) The GotIt! proxy has no way of knowing which pages the user will actually hit so it downloads all of them that are linked to. It undoubtedly filters out these links for it's own use during the action of serving the linking page to the user. Depending on the performance of serving machine and the nature of the linking page, this could actually slow down performance if, for example, the user is only interested in the 3,4, 6 and 9th links as all 1 to 12 links will be fetched. GotIt's pre-fetch model blindly prefetches all links on a page. This would result in too many prefetched pages for the average browser set up and platform as sites now commonly have many navigation links at their top levels.

SUMMARY OF INVENTION

According to one aspect of the invention there is provided a method of browsing an HTML page on a TCP/IP network comprising: loading a first HTML page into browser memory from a network server; loading said memory into display memory for displaying said the page; providing means for selecting multiple URLs on the HTML page; and preloading pages corresponding to user selected URLs into browser memory without loading them into display memory or affecting the ability to select further URLs. The solution allows multiple requests by the user and results in a much reduced start-to-finish time for a user who is using the net in a 'query, list, brows ' mode (as is commonly done). The user need not look at every page downloaded and available from cache memory. As most of the pages will be 'instant' from the cache, page providers may benefit too by

receiving more 'hits' as users are much more likely to view a page if they can see that it will be displayed 'instantly' from the browser cache.

Advantageously, this aspect of the invention further provides a step of on selection of a URL and before preloading the corresponding page, updating a navigation list with the URL. This allows a user, having finished making multiple selections from the first page, to instruct the browser to display pages from the navigation list even though the pages may not be fully loaded into memory. More advantageously the method further comprises updating the navigation list with information as to whether a URL on the list is fully loaded. In which case the user can maximise the time spent in viewing only fully loaded pages.

Advantageously, on selection of a URL and before preloading the corresponding page into cache memory, there is provided a further step of updating a cache index file with the location of the page in cache. This allows a user, having finished making multiple selections from the first page, to instruct the browser to display pages from the navigation list even though the pages may not be fully loaded into memory.

Advantageously pages corresponding to selected URLs are not displayed during preloading so that the user may view the original page until finished checking all the URLs. After the user is finished checking he may move on to one of the selected web pages.

One beneficial method for navigating the selected URLs is to build a list of page references including that of the first page. A drop down list may be provided for selecting a page reference in the list so that the corresponding page can be displayed. Alternatively or as well as 'forward' and 'back' GUI buttons may be provided for selecting the previous or next page reference in the list and displaying the corresponding page.

The user is already used to using control-mouse-left-click (for copy-drag actions in Windows) and the above solves the problem in an intuitive way. As the model suggests, making use of the 'forward-arrow' for navigation through the pages downloaded in the background is a familiar navigation method for the user. The browser cache will end up in exactly the same state it would if he/she had interactively downloaded the pages one at a time and then used 'back-arrow' to return to the index page. Moreover the user is not faced with having to continually jump back to the top level and has much less 'waiting time' overall as the downloads occur in parallel. Furthermore the user has something active to do/read while downloads are going on the background the operation will be even more efficient/enjoyable.

Preferably a page reference is added to the list when its corresponding URL is selected from the first page so that the list is order sequentially in a logical order. Alternatively a page reference maybe added to the list when its corresponding page is substantially loaded into browser memory so that if a user moves forward in the list there is greater chance that the page will have loaded into the browser. This solution provides for a much more easily navigable model for doing parallel downloads over the current possible mechanisms. The solution is coherent with the current user model of path walking with marks for places visited. The solution is applicable to a 'usage scenario' employed many times every day in a highly competitive area and is easily detectable.

Browser memory comprises cache memory. When a preloaded page in proxy memory is required for display on a browser, the browser sends a TCP/IP request to the proxy for a page stored in proxy memory. The page is copied into both cache memory and display memory for display on the browser. It is advantageous to preload selected pages directly into cache memory from an TCP/IP network so that a quicker internal request from the browser to the cache memory renders the displayed page to the user in a shorter time as opposed to a slower TCP/IP request from the browser to the proxy.

According to another aspect of the invention there is provided a system for browsing an HTML page on a TCP/IP network comprising: means for loading a first HTML page into browser memory from a network server; means for loading said memory into display memory for displaying said page; means for selecting multiple URLs on the HTML page; and means for preloading pages corresponding to selected URLs into browser memory without loading them into display memory.

According to another aspect of the invention there is provided a method of processing a URL database comprising: loading the HTML result of a URL search into browser memory; loading the HTML results into display memory and displaying a list of located URLs; providing means for selecting multiple URLs from the list; and preloading web pages corresponding to selected URLs without displaying said web pages.

According to a further aspect of the invention there is provided a computer program product comprising a computer usable medium having computer readable program code means embodied in the medium for processing work items in a data processing system, the program code means comprising: code means for causing the data processing system to load a first HTML page into browser memory from an network server; code means for causing the data processing system to load said first HTML page into display memory for displaying said page; code means for causing the data

processing system to provide means for selecting multiple URLs on the HTML page; and code means for causing the data processing system to preload pages corresponding to selected URLs into browser memory without loading them into display memory.

According to a further aspect of the invention there is provided a computer program product comprising a computer usable medium having a computer readable code page embodied in the medium for processing work items in a data processing system, the code page comprising: code means for instructing a code page browser to display one or more links to other codes pages; and code means for, on selection of one or more of the displayed links, instructing a code page browser to load one or more of the code page into browser memory whilst instructing the code page browser to retain the current code page display of said one or more links.

Once a user has completed the process of viewing all his/her desired pages by initiating each download interactively, all the previous pages are easily selectable for reviewing using the 'back' and 'forward' features of the browser. As these work in conjunction with the browser cache one can quickly move backward through the list (and subsequently forward again). If it were possible to reverse this process how much easier say Yahoo or patent searches would be.

Although the invention is described in terms of TCP/IP networks it could equally apply to other types of packet networks. Furthermore although the invention is described in terms of HTML pages it could equally apply to other types of mark up language data containing URLs. Although the invention is described in terms of URLs it could equally apply to other network data references.

BRIEF DESCRIPTION OF DRAWINGS

In order to promote a fuller understanding of this and other aspects of the present invention, an embodiment will now be described, by way of example only, with reference to the accompanying drawings in which:

Figure 1 is a schematic representation of a computer system of the embodiment;

Figure 2 is a screen dump of a cache memory;

Figure 3A and 3B represent the procedures of the embodiment; and

Figure 4 is a screen dump illustrating one of the advantages of the embodiment.

DETAILED DESCRIPTION OF PREFERRED EMBODIMENTS

Referring to Figure 1 there is represented an operational platform 10 connected to a plurality of servers 12A, 12B, 12C, via a proxy server 13 and an Internet network 14. Platform 10 is an Intel Pentium II 450Mhz processor based computer, having 128 MB 100 MHz SDRAM (reference number 16 on Figure 1), 17 GB ATA-33 hard drive (also represented as reference number 16 on Figure 1), 19" colour SVGA monitor, 16 MB video memory (reference number 18 on Figure 1), 17/40 x CD-ROM drive, 3 PCI, 1 ISA and 1 AGP expansion slots and Microsoft Windows 98 operating system. The platform 10 has a TCP/IP browser 20 such as Netscape Navigator. The servers 12,13 have similar specifications to platform 10 but run under Microsoft Windows NT operating system. Although the example platform is described in detail, any processor and operating system combination capable of running a TCP/IP browser is suitable. The example browser need not be Netscape Navigator but any browser capable of receiving mark up code pages from a network, for instance Microsoft Internet Explorer.

Browser 20 is the client application which is used to view pages of information sent by the servers. Five components of the browser are of interest in the present embodiment. The browser code 22; the cache memory 24; the cache index 26; and the navigation list 28 are known components of browser 20. In this embodiment the invention is centred on a browser plug-in 30 so that no adaptation of the browser code is necessary and the invention may be incorporated into any present browser set up by adding the component to the browser.

In another embodiment the browser code 22 may be modified with necessary functionality so that no plug-in to the shipped browser is necessary.

In a further embodiment the methods of the invention may take the form of code (such as Javascript) within the web page data instructing the browser to carry out the steps of the invention.

Proxy server 13 is the gateway to the Internet for the platform 10 and potentially many other platforms. When web page information is downloaded from servers on the Internet to the browser it is stored in the proxy server 13 as well as being transferred to the cache memory 24 of the browser 20. If any other browser using the proxy server 13 as a gateway requests the same page then the proxy server 13 can transfer it from its own memory without incurring the delay by transferring it again from the Internet. The proxy also has some security functionality for limiting what information is sent and received. Internal web pages can not be sent through 'fire wall' security measures in the proxy server.

Browser code 22 contains the functionality of the browser including the TCP/IP interface code and the mark up language interpreter (both not

shown). Cache memory 24 stores several pages 32A and 32B including main pages and related files of information from the Internet servers. The proxy server 13 has a similar cache memory for Internet information but stores more information than a browser cache memory so that it may supply the other connected browsers. Cache index 26 contains the groups 34A,B,C of names and references of the files that make up web pages 32A and 32B. Navigation list 28 is a sequential list of web pages visited in the browser session with associated URLs. Both Netscape Navigator and Microsoft Internet Explorer have cache memories 24 where HTML, GIFs, MP3 etc. files are cached in a hard disk directory with a cache index 26. With Internet Explorer these are held at C:\winnt\profiles\<user>\Temporary Internet Files\index.dat and in Netscape the equivalent file is, for example, C:\Program Files\Netscape\Users\<user>\Cache\fat.db. For Internet Explorer, these can be seen using Windows Explorer as shown in Figure 2.

The navigation list contains the URLs 36A,B,C,D visited including the present web page 36B (indicated by the bold outline), those previously visited 36C,D (indicated by the dotted outlines) and the page 36A to be visited in the future (indicated by the dotted line). The cache index 26 holds file references 34A for the present web page and perhaps those of the previous web page e.g. 34B and 34C but not the web page references of 36D which have been removed due to limited space for illustration only. The actual embodiment has extended cache memory 24 and holds from two to ten or more complete web pages. The cache memory 24 holds the web page data 32A and 32B as referenced by the cache index 34A anbd 34B respectively and some or all of previous web pages, for example those referenced by pages 36C and 36D. The mark up code interpreter in the browser code 22 uses the mark up files 32A as indexed by cache index 34A and pointed to by list item 36A to represent a web page 38 in graphical form in the display memory 18 which is displayed to the user on the monitor.

Plug-in 30 uses a defined way of extending the functionality of the well-known browser described above without having to alter the underlying browser or requiring a recompilation of it. It works by a convention of entry points (or functions) that the browser can call within the plug-in and vice versa. The functions are called at significant points in the processing of a downloaded page and different page 'types' can be associated with different plugins. It is implemented as a dynamically loaded library and thus has access to the underlying computer's file system and other facilities (and so could manipulate the browser cache), it also has the ability to intercept mouse and keyboard actions from the user and have visibility of the browser's URL fetching via the defined browser to plug-in interface. Plugins conventionally deal with new file types (such as the well known VRML plug in that displays virtual reality

markup language and interacts with the user to control the display). However plug-in 30 deals with existing file types such as .HTML with slightly different processing than is conventionally used.

To implement the embodiment we need to achieve the following functional units as identified in Figure 1: means 40 to register a plug-in that is called for 'standard' HTML and HTM web pages; means 42 to capture mouse or keyboard events to control the page downloads; means 44 to request that pages are downloaded without being displayed and furthermore that these pages can be placed in the Netscape cache; means 46 to maintain a list of pages that have been so placed in the cache to enable them to be displayed in response to the user selecting the 'forward' and 'reverse' option in the user interface; and means 48 to trigger Netscape to display a particular page that is stored in the Netscape cache.

Each of these five parts of the design is implemented using the Netscape plug-in interface and will be discussed in turn.

1) Registering a plug-in that is called for 'standard' HTML and HTM web pages.

When Communicator starts up, it checks for plug-in modules in the plugins directory for the platform:

MS Windows: plugins subdirectory, in the same directory as the Communicator application.

Mac OS: Plug-ins folder. A Mac OS plug-in can reside in a different directory if one installs a Macintosh alias that links to the plug-in in the Plug-ins folder.

UNIX: usr/local/lib/Netscape/plugins or $HOME/.Netscape/plugins. If a different directory is wanted, the NPX_PLUG-IN_PATH environment variable is set to its filepath, for example, $HOME/yourplugins:/ usr/local/lib/Netscape/plugins. Communicator searches any directory that this variable specifies. The local user location, if it exists, overrides the network location.

On all platforms, the plugins default subdirectory or folder is in the same directory as the Communicator application. Users can install plugins in this directory manually, by using either an installation program or Communicator's JAR Manager facility. The plugins currently installed can be viewed under Netscape by selecting the 'help' menu option and the "About Plug-ins" menu item. This gives a list of installed plugs ins currently registered with the Netscape Browser and

identifies the 'default' plug in that is run for all file types not registered with another plug in:

Netscape Default Plug-in
File name: C:\PROGRA~1\Netscape\COMMUN~1\Program\plugins\npnul32.DLL

Default Plug-in

Mime Type	Description	Suffixes	Enabled
*	Netscape Default Plug-in	*	Yes

There is usually no other plug in that is registered for the file type .HTM or .HTML as this type being handled by the default services of Netscape. The plug in architecture does not prevent this happening and allows a plug-in that is activated when a file of type .HTM or .HTML is loaded. A dynamic link library (DLL) file is developed that conforms to the plug-in conventions as documented in the "Netscape Communicator Plug-in Guide" with a number of known function entry points. The DLL file is placed in the appropriate directory on platform 10. When it starts up, Communicator checks for plug-in modules in the plug-in directory for the platform and registers them. It determines which plugins are installed and which types they support through a combination of user preferences that are private to Communicator and the contents of the plugins directory. When Netscape is initialising it sweeps this directory and for each plug-in DLL that it finds it calls the NPP_Initialise function entry point in the DLL file. It also examines the DLL's version information.

On Windows, the plugin's directory is located in the same directory as the Communicator application and has a 8.3 filename beginning with NP and ending with .DLL. The Windows version information for the plug-in DLL determines the MIME types, file extensions, file open template, plug-in name, and description. In the MIME types and file extensions strings, multiple types and extensions are separated by the "|" character, for example: video/quicktime|audio/aiff|image/jpeg

The version stamp of the plug-in DLL contains the following lines so that Communicator recognises the plug-in,:

File Extents: for file extensions

MIME Type: for MIME types

Language: for language in use

In order to display an example of this use Windows Explorer to get to the C:\Program Files\Netscape\ Communicator\Program\Plugins> directory

and right click on one of the np*.DLL files - use the pop-up menu to view the DLL's properties. Up will come a notebook, click on the Version Page Tab -> You will see fields called "File Extents" and "MIME Type" these fields are decoded by Netscape using the convention above.

By creating a Netscape plug in using the instructions in the plug-in SDK and setting the values in the plug-in DLL's properties information to indicate that this plug is for file extensions of .HTM and .HTML we will cause Netscape to communicate with this plug in when files of this type are loaded.

2) Capturing mouse or keyboard events to control mouse or keyboard events to control web page downloads.

In order to capture mouse events that are driven from the HTML page the plug-in 30 operates as a 'full page' plug-in. A full page plug-in is visible but not part of an HTML page. The server looks for the media (MIME) type registered by a plug-in, based on the file extension, and starts sending the file to the browser. Communicator looks up the MIME type and loads the plug-in if it finds a plug-in registered to that type. This type of plug-in completely fills the Communicator page and are commonly used for document viewers, such as Adobe Acrobat. Plug-in 30 receives mouse events that take place while the mouse pointer is over it's own display window. The following events are visible to the plug-in

WM_PAINT
WM_LBUTTONDOWN
WM_LBUTTONUP
WM_LBUTTONDBLCLK
WM_RBUTTONDOWN
WM_RBUTTONUP
WM_RBUTTONDBLCLK
WM_MBUTTONDOWN
WM_MBUTTONUP
WM_MBUTTONDBLCLK
WM_MOUSEMOVE
WM_KEYUP
WM_KEYDOWN
WM_SETCURSOR
WM_SETFOCUS
WM_KILLFOCUS

These events will be passed to the plug-in via the "int16 NPP_HandleEvent(NPP instance, NPEvent *event);" function entry point defined in the plugin's DLL. In the embodiment the user holds down the control key to indicate that a background download of a URL. This causes

a WM_KEYDOWN event to take place for the control key. Plug-in 30 maintains a flag that represents whether the control key is down or up.

3) Requesting pages for download in cache memory without displaying

When the download of a new page is requested and a plug-in is registered for that page type then browser 20 will create a new 'stream' that represents the stream of data in the web page. To tell the plug-in when a new stream is created, browser 20 calls a NPP_NewStream method. This method also determines which mode it should use to send data to the plug-in via a return parameter in the function call.

The NPP_NewStream method has the following syntax:

NPError NPP_NewStream(NPP instance, NPMIMEType type, NPStream *stream, NPBool seekable, uint16* stype);

The instance parameter refers to the plug-in instance receiving the stream; the type parameter represents the stream's MIME type. The stream parameter is a pointer to the new stream, which is valid until the stream is destroyed. The seekable parameter specifies whether the stream is seekable (true) or not (false). Seekable streams support random access (for example, local files or HTTP servers that support byte-range requests). The plug-in can set the type output parameter type to various transmission modes and the embodiment chooses the mode based on whether the control key is down or not. If the control key is currently down (background download is requested) the type NP_ASFILEONLY is selected. The plug-in gets full random access to the data using platform-specific file operations. Browser 20 saves stream data to a local file in the cache, and, when the stream is complete and delivers the path of the file to the plug-in through a call to NPP_StreamAsFile. The URL will not be launched in a browser window or replace the contents of the current window. This mode will fetch the page to the Netscape cache without causing it to be displayed on a window. If the control key is not held down then the type of the fetch will be NP_NORMAL which will allow for the default behaviour of the page download.

4) Maintaining a list of pages placed in the cache to enable display in response to the user selecting the 'forward' and 'reverse' option in the user.

Plug-in 30 has visibility of all HTML pages requested, those downloaded in the background and their cache filenames. By monitoring the requests, downloads and file names the plug-in can maintain a sequential list of web pages and URLs in the order they were requested.

5) Triggering the display of a particular page.

As Netscape will have populated its cache for those pages subject to a NP_ASFILEONLY type of download plug-in 30 responds to the chosen user interface action (e.g. WM_KEYDOWN on < Control > and < -> >) to make the call back into the browser to display the next page using the NPN_GetURL function (defined below).

```
NPError NPN_GetURL(NPP instance,
                   const char* url,
                   const char* target);
```

NPN_GetURL is used to request browser 20 to load a URL into a particular HTML window or frame for display, or to deliver the data of that URL to the plug-in instance in a new stream. When the browser attempts to display the URL it will check it's cache as usual and because the page has already been downloaded to the cache it will find it there.

Operation of the plug-in

Using the plug-in 30 to preload selected web pages, the user may enhance present web searches as now described.

A) user enters search criteria.

B) user is presented with list of results.

C) using the plug-in selection mechanism (for instance holding the control key while selecting an item) the user can select a number of links while the browser view remains focused on the list of results and the browser stays 'live' to accept further requests.

D) as the requested URLs arrive at the browser the 'forward-arrow' of the browser stops being greyed out, indicating that there is a page present in the cache memory 24 'forward' from the present web page.

E) The user can continue making selections or make use of the forward arrow once finished. The 'forward-arrow' will grey whenever the user is at the latest requested page present in the cache and ungrey as any new 'background upload' pages arrive.

Referring to Figure 3A a more detailed description of the operation of the plug-in is given. On start up of the platform the operating system, browser and plug-in are loaded into operational memory from disk and initialised. In the course of Internet browsing the user is presented with an HTML screen containing multiple URLs from for example a search or a bookmark list. The user selects a URL (step 100). The browser checks whether the web page is already in cache memory (step 102) and displays the page if present (step 104). If not so present the plug-in checks for a background select (or preload select) (step 106), adds the URL (step 108) to the navigation list 28 and requests preloading of the page (step

110) into cache memory 24. If the user selects a foreground load (at step 106) then the browser proceeds as known by requesting the page at the same time as it is displayed (step 112).

After a URL is loaded (see Figure 3A) into the navigation list without having completely loaded the web page information into the cache memory it is possible to move from page to page by selecting 'Backwards' or 'Forwards' icons or options. In the present embodiment the cache memory is of a size to allow several pages of web page information, it is finite and web page information for previous links will eventually be overwritten. Hence 'navigation' through the list may require reloading of pages.

Referring to Figure 3B the process of web page information arriving at a web browser is depicted. Once a URL is pre-selected the navigation list is directly updated with the URL (108) and the web page information is received by the browser (step 114). It is then saved to the cache memory (step 116). Once all the information for a page is received icons are updated (step 118). Forward button 39B on the browser is normally grey indicating that there is no forward page to move to, when the page is pre-selected it is ungreyed to indicate that there is a forward page to move to. It may be ungreyed when the web page is fully loaded to indicate a fully loaded page or alternatively when the page is preselected to indicated a new page forward. The cache index is updated (step 120) as the web page information is added to the cache. An additional feature to the embodiment, a cached page table, is updated (step 122) with details of complete cached pages. This allows the user to jump ahead to fully received pages in the navigation list immediately from a pull down list of the pages with fully received pages indicated (for example by highlighting). The cache page table is a flag set in the navigation list 28 (e.g. by hitting the forward button 31A).

The following is an illustration of prefetching according to the embodiment. A user is browsing page 38 (see Figure 1) which web information is displayed from the data indicated by 32B (indicated in Figure 1 by darker lines) in cache memory 24. The web data files 32B are referenced in cache index reference 34B and this index is indicated by URL 36B (all indicated by darker lines in Figure 1) in the navigation list. The user may pre-select a new URL (step 100) and the URL 36A is added to the navigation list (step 108) and the web page information requested (step 110) and loaded into cache at 32A. An indication of when the web page is fully loaded is made by ungreying the Forward button 39B but the web information 32A is not displayed until the user selects the URL 36A in the navigation list 28.

The known browser feature that allows one to right-click on the forward and back arrows to see what pages are in the ring would be clearly useful. The browser could easily use this to show files requested and not yet downloaded in a manner that fits naturally with the current behaviour. This option is displayed in Figure 4 (the user right clicked on the Forward button of Internet Explorer). When one right clicks on the 'forward' or 'back' buttons, one is presented with a list of sites one has visited. These sites are already cached and will display immediately. In the envisioned embodiment of this idea the forward button would display a list of sites logically 'forward' in the users navigation path, these would include sites as today and sites requested with the background cache file download mechanism that have already arrived and are available in highlighted (or darker) text and sites that have been requested for background cache download that have not yet fully arrived in greyed out text.

In summary, this specification relates to the Internet world wide web page downloading and in particular relates to browser prefetching of web pages. There is disclosed a method and system of browsing an HTML page on the Internet comprising preloading pages corresponding to selected URLs into browser cache memory without loading them into display memory or effecting the ability to select further URLs. The method loads a first HTML page into browser cache memory from a network server via a proxy server and the Internet and loads said memory into display memory for displaying said the page. A user may then select multiple URLs on the HTML page without causing them to be displayed but causing then to be loaded as selected or on mass into cache memory not display memory or proxy.

Now that the invention has been described by way of a preferred embodiment, various modifications and improvements will occur to those person skilled in the art. Therefore it should be understood that the preferred embodiment has been provided as an example and not as a limitation.

CLAIMS

1. A method of browsing a "markup language" browser page from a network comprising:

loading a first "markup language" page into cache memory from a network server;

loading said page into display memory for displaying said the page;

providing means for user selecting multiple URLs on the "markup language" page; and

preloading pages corresponding to selected URLs into browser cache memory from at least one network server.

2. A method according to claim 1 further comprising, on selection of a URL and before preloading the corresponding page, updating a navigation list with the URL.

3. A method according to claim 2 further comprises updating the navigation list with information as to whether a URL on the list is fully loaded.

4. A method according to claim 1,2 or 3 further comprising, on selection of a URL and before preloading the corresponding page into cache memory, updating a cache index file with the location of the page in cache.

5. A method according any of claims 1 to 4 wherein pages corresponding to selected URLs are not displayed during preloading so that the user may view the original page until finished checking all the URLs.

6. A method according to claim 2 further comprising providing a drop down navigation list for selecting a page reference in the list so that the corresponding page can be displayed.

7. A method according to claim 6 further comprising providing 'forward' and 'back' GUI buttons for selecting the previous or next page reference in the list and displaying the corresponding page.

8. A method according to any one of claim 1 to 7 further comprising preloading selected pages directly into cache memory from an TCP/IP network

9. A system for browsing an HTML page on a TCP/IP network comprising:

means for loading a first HTML page into browser memory from a network server;

means for loading said memory into display memory for displaying said page;

means for selecting multiple URLs on the HTML page; and

means for preloading pages corresponding to selected URLs into browser memory without loading them into display memory.

10. A computer program product comprising a computer usable medium having computer readable program code means embodied in the medium for processing work items in a data processing system, the program code means comprising:

code means for causing the data processing system to load a first HTML page into browser memory from an network server;

code means for causing the data processing system to load said first HTML page into display memory for displaying said page;

code means for causing the data processing system to provide means for selecting multiple URLs on the HTML page; and

code means for causing the data processing system to preload pages corresponding to selected URLs into browser memory without loading them into display memory.

The Patent Office

INVESTOR IN PEOPLE

Application No: GB 9911739.2 **Examiner:** Julyan Elbro
Claims searched: 1 to 10 **Date of search:** 21 January 2000

Patents Act 1977
Search Report under Section 17

Databases searched:

UK Patent Office collections, including GB, EP, WO & US patent specifications, in:

UK Cl (Ed.R): G4A (AMU, AUDB)

Int Cl (Ed.7): G06F 17/30

Other: ONLINE: COMPUTER EPODOC JAPIO WPI Selected Internet sites

Documents considered to be relevant:

Category	Identity of document and relevant passage	Relevant to claims
X, Y	WO 97/46955 A1 AT&T see abstract and fig. 3.	X: 1, 4, 5, 8 to 10 Y: 2, 3, 6, 7
Y	HCLU, "Netscape Navigator", published 1996, available from http://libsun1.jr2.ox.ac.uk/training/netscape.html see especially the section "Back and Forward".	2, 3, 6, 7

X	Document indicating lack of novelty or inventive step	A	Document indicating technological background and/or state of the art.
Y	Document indicating lack of inventive step if combined with one or more other documents of same category.	P	Document published on or after the declared priority date but before the filing date of this invention.
&	Member of the same patent family	E	Patent document published on or after, but with priority date earlier than, the filing date of this application.